THE USE OF FRAMES IN DATABASE MODELING

by

BARBARA MOORE SWEET

B. S., Kansas State University, 1982

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1984

Approved by

Roger T. Hartley
Major Professor

TABLE OF CONTENTS

PAGE
LIST OF FIGURES iii
ACKNOWLEDGEMENTS iv
DEDICATION v

PREFACE 1

CHAPTER 1
INTRODUCTION 4
1.1 Introductory remarks 4
1.2 Desirable Characteristics of Databases 5
1.3 Introducing New Database Models in Industry 7
1.4 Organization of Thesis 8

CHAPTER 2
THE CONCEPT AND CAPABILITIES OF FRAMES 10
2.1 Background 10
2.2 A Brief History of Frames 11
2.3 FRL Terminology 13
2.4 The Frame System Structure 16
2.4.1 The FDM Attribute Frame 16
2.4.2 The FDM Element Frame 19
2.4.3 Using Frames to Represent Frames 20
2.5 The Conceptual Capabilities of Frames 21
2.5.1 Inheritance 21
2.5.2 Default Values 26
2.5.3 Demons 26
2.5.4 Perspectives 29
2.5.4.1 Perspectives in KRL 32
2.6 The FDM User-View Frame 32
2.6.1 Security 34
2.6.2 The Specification of User Views 35
2.7 The FDM Inference Frame 38

CHAPTER 3
ORGANIZATIONAL ISSUES 40
3.1 Multivalued Dependencies 40
3.1.1 Multivalued Dependencies in the Network, Hierarchic, and Relational Models 41
3.2 Multivalued Dependencies in the FDM 50

CHAPTER 4
ACCESS ISSUES 52
4.1 Multiple User-Views 52
4.2 Representation of Incomplete Information 54
4.3 Update and Deletion Anomalies 57
4.3.1 Anomalies in the Network, Hierarchic, and Relational Models 57
4.3.2 Update and Deletion Anomalies in the FDM 62

CHAPTER 5
QUERY ISSUES 64
5.1 Unprocessable Queries 64
5.2 Query Optimization Using Semantic Constraints 66
5.2.1 Two Methods of Semantic Query Optimization 68
5.2.2 Semantic Query Optimization in the FDM 69

CHAPTER 6
THE SEMANTIC DATA MODEL 71
6.1 Components of the SDM 71
6.2 A Criticism of the SDM 72

CONCLUSION 75

BIBLIOGRAPHY 77

LIST OF FIGURES

FIGURE		PAGE
1	FRL Frame Structure	15
2	Frame Structure Example	18
3	FDM Frame Types	22
4	FDM Element Frame Examples	24
	FDM Attribute Frame Examples	25
5	FDM Frames with Demons	28
6	Perspective Tree	31
7	Perspective Matching Example	33
8	FDM User-view Frame Structure	36
9	N-M and I-M Dependencies in the Network and Hierarchic Models	42
10	Unnormalized Relation Example	45
10.1	Fourth Normal Form of Figure 10	45
10.1a	Additional Relation to Figure 10.1	45
11.1	Cross Product of Figure 10	47
11.2	Disjoint Representation with Null Values of Figure 10	47
11.3	Minimal Number of Values with Null Values of Figure 10	47
11.4	Minimal Number of Values with Repetitions of Figure 10	49
12	INSTRUCTORS Relation Example	59
13	FACULTY, COURSE, COURSE-STATS Relation Examples	60
14	Semantic Constraint Example	67

ACKNOWLEDGEMENTS

I wish to acknowledge with deep appreciation the members of my thesis committee for their encouragement and advice.

My thanks to Dr. Roger T. Hartley, who acted as my major professor and committee chair. Dr. Hartley gave me reassurance and guidance during the few short months I had to complete this work.

I also wish to thank Dr. Elizabeth A. Unger, who was my first advisor at Kansas State University and did me the honor of being on my committee. Dr. Unger told me in 1979 that I would go on to graduate studies, "...probably in Artificial Intelligence."

Finally, my thanks to Dr. Rodney M. Bates, who was of more help while on my committee than he might realize. Dr. Bates gave the objective ear which kept this thesis from reading like a chapter out of Alice in Wonderland.

DEDICATION

To Robert: husband, best friend, and the <u>most</u>
patient man I've ever met.

PREFACE

In all areas of computer science there is a demand from industry to create new tools for information management. The amount of information which computer system users need to have available and well managed is not, and will not, be decreasing. This need is causing the development of better hardware, better software design techniques, improved ways to link hardware and software to make communication and processing more easily available, and better methods to store and use information. At this time it seems impossible to keep up with current demands, much less to plan for future needs. But because the demands will continue to increase faster than they can be met, it is essential for researchers in computer science to design systems now that can fill the growing demands in the future. This is as true for the field of database modeling as any other area of computer science.

An analogy can be drawn to a parent purchasing clothing for a child. Since the parent knows the child will continue to grow in the future, the "wise buyer" looks for a fit a bit larger and longer than the child's current size. This gives the new garment a longer period of use.

It is even more important for database models to be flexible so that they can cope with the growth trends and shifting priorities of industrial consumers. An example similar to the purchase of clothing for a growing child is the manner in which a woman might look for a dress to buy. She may look for a style and fabric which can effectively allow for future shortening or lengthening of the hem. This way, as fashions change, the dress can be altered so as to be current rather than being discarded or being inappropriately or unfashionably used.

"Smart" consumers are constantly using strategies to make purchases today which will still fill the needs of the future. They buy homes with extra space if family growth is expected. The same is true of automobiles, cold-storage appliances, and other long lasting items. Many of these items are what are called "big ticket" items - expensive. This accounts for future minded buying strategies.

Commercial databases are certainly "big ticket" items. Unfortunately, many database systems are purchased to fill immediate needs and may be found inadequate when the future becomes the present. It is not hard to understand why industry continues to purchase the "good old reliable" database models, even though they are inflexible and constraining to users. They may not supply the future needs of the users, but new models are hard to use, still not very user friendly, and really don't offer any greater range of user capabilities.

It is important that database designs be developed which anticipate the needs of the future but do not increase in user complexity and design inflexibility at the same time. There are many available methods to make databases reliable and consistent, but they also require industrial consumers to accept greater degrees of adjustment in adapting them for use.

Databases must be designed to include features which make them flexible, easy to use, and reliable. Users should not need to know how the database is organized, what processes are required to locate and manipulate specific contents, or how to optimize interaction with the database. The database should handle these things. The database must be a partner to the user, not an obstacle, in information management. As the size of databases and the uses they must accommodate continue to

increase, it will become increasingly unrealistic to expect those who interact with databases to be able to keep track of the rules and requirements of their use. The databases must perform this task and be adaptable to changes and growth in the application areas they are being used for. If not, why should industrial consumers accept improved database models?

The frame-based model proposed in this thesis is intended to supply both a reliable and consistent means of information management and a model which would support more complex user needs, but not more complex user responsibility. By incorporating the semantics of the application environment into the database itself the model has the capability of monitoring and controlling its use, and to supply a conducive environment for user interaction.

CHAPTER 1
INTRODUCTION

This thesis introduces a database model based on frames, a conceptual structure developed in Artificial Intelligence knowledge representation. This frame based database, hereafter referred to as FDM (Frame Database Model), is offered as an alternative database model for commercial applications.

The body of the thesis will argue the importance of alternative methods of data base modeling, the appropriateness of frames as structures for database modeling, and describe the benefits available to commercial applications from a frame based model. The next section will explain the approach and organization of this thesis.

1.1 INTRODUCTORY REMARKS

The approach this thesis takes concerns the intentional vs. the extensional perspective of database modeling. There is no discussion of issues which involve the physical or implementation level of database development and maintenance, nor will specific issues of data organization and run time efficiency be addressed. The effect of the conversion from conceptual model to logical model will be discussed as necessary, but it is not in any way the focus of this thesis.

It should be stated that an FDM will probably have a slower processing time than standard systems because of the increased complexity. This is admittedly an undesirable side effect in commercial application databases since processing turn around time is usually a high priority. The problem is unavoidable but could be diminished or

eliminated by application of good software engineering techniques and by future developments in firm/hardware. Since run time efficiency is a problem, in varying degrees, with all database implementations, it might be the case that the capabilities of an FDM outweigh this disadvantage.

1.2 DESIRABLE CHARACTERISTICS OF DATABASES

It is necessary to establish the characteristics which are desirable in any database. In addition, it will be demonstrated that the current and future needs of commercial applications will suggest that the development of new database models is a timely research area.

Within the application area of an implemented database the data elements of that database are tokens which have semantic reference to "things" in the real world. These data elements are the contents of the database. As stated by Hammer [21], at any given moment the current state, or contents, of a database should capture the current state of its application environment. However, the contents of the database have no representation of intentional meaning. It is necessary for a user of the database to apply processes to the database such that process results are usable and meaningful.

A database representation of an application environment should also be natural and unrestricted [2]. Many database models are designed to be as natural and unrestricted as possible. They are also limited by the necessity of protecting the integrity and reliability of the contents of an instance of an implemented database. The different approaches of database model design, structured (network and hierarchic) and unstructured (relational), are used to create models which strictly

adhere to rules that minimize erroneous information. Examples of such rules include the limitations on representing many-to-many multivalued dependencies (all standard models), and the representation of incomplete information (relational model).

Because of the nature of the two different approaches, each design method has priorities of organization and manipulation of the contents of a database. For example, the effect that different levels of normalization have on the organization of relational databases. In some cases, such as in the Boyce-Codd normal form, the organization of the database is effected in a restricted fashion such that it may not be possible to retain real world relationships among attributes.

It is not enough for a database model to be designed so that an implementation has structural or organizational restrictions which comply with the semantic references of the outside application environment. The database model should also contain representations of those semantic references.

Some of the problems associated with developing databases with the aforementioned desirable characteristics are caused by the database model. However, many of the problems which affect the objective of a natural and unrestricted representation occur when the conversion is made from the conceptual model of a specific database to the actual implementation. The common database models (network, hierarchical, relational) and even the more recently developed semantic data model, must be implemented using data definition languages. Different data definition languages may impose their own limitations on the representation of the database. An example of this is the use of virtual records in the CODASYL implementation of the hierarchic model.

1.3 INTRODUCING NEW DATA BASE MODELS IN INDUSTRY

It can be argued that the current database models are already able to accommodate the needs of database users in industry. Improvements brought about by research are often not readily accepted for use in industry. In addition, industry is reluctant to adopt improvements in database modeling due to the need for retraining and adjustment to the use of new models when implemented.

There are two reasons for the importance of research in the area of "intelligent" database models, models which can monitor and/or intercede in the organization and access of themselves, for industry.

First, a major problem with current databases is that a user must have some degree of understanding of the implemented database, the manipulations which may be performed on the database, and the rules for using these manipulations. A user will want a database to be as easy to understand as possible. Users also do not want to have to relearn or adapt to the changes in the use of a database as research develops modifications in modeling and design. The solution for the user is to be able to work with an intelligent database. Such a database could greatly reduce the necessity for the user to understand the database itself and the rules for its use.

Second, in the future, the maintenance of integrity in a given database will be increasingly important, as the size of databases increases, and the practice of single database use by multiple users for multiple applications expands. It will be increasingly important under such conditions for any given user of a database to be unable to corrupt that database for other users. Data bases of the future should be able to monitor themselves to be certain they are safely used and maintained.

This is not to say that a database should or will become a "black box". The database would be understandable by those who are in a position to need to know about the workings and organization of the database (the database administrator, DBA). In fact, work on the XPLAIN expert system [20] indicates that it is possible for the database to respond to a query of why it has performed in a given manner using reasoning paths, rather than responding using a trace-back of the processing execution. However, the user would not need an indepth understanding of the way the database functions. The user would only need to know information essential to interaction with the database with regard to their particular application area.

1.4 ORGANIZATION OF THESIS

Chapter 2 explains the capabilities and structure of frames. This chapter also contains a brief history of the use of frames in Artificial Intelligence knowledge representation.

Chapter 3 discusses the issue of multivalued dependencies. The methods in which the standard database models treat this issue are discussed. The way in which the FDM is affected by multivalued dependencies, or in point of fact is not affected, is presented.

Chapter 4 discusses issues and problems which arise in database access and how the FDM might resolve them. The issues and problems include multiple user view facilities, representation of incomplete information, and update and deletion anomalies, and query processing.

Chapter 5 discusses database query issues. There are two topics included in this chapter: unprocessable queries and semantic query optimization.

Chapter 6 concludes this thesis. In addition to concluding remarks, this chapter contains a discussion of the SDM database model, including an explanation of its components and a criticism of its limitations.

CHAPTER 2
THE CONCEPT AND CAPABILITIES OF FRAMES

Chapter 2 explains the concept and capabilities of frames, and discusses the structure and components of the FDM. Section 2.1 gives a brief background to the use of frames for knowledge-bases in Artificial Intelligence knowledge representation. Section 2.2 contains a brief history of the origins of frames as an Artificial Intelligence construct. Section 2.3 offers the FRL (Frame Representation Language) terminology and form for use in this thesis to facilitate the presentation of examples. Section 2.4 explains frame system structure, and introduces the FDM attribute frame (Section 2.4.1) and element frame (Section 2.4.2) types. Section 2.4.3 discusses the use of frames to represent and define frames.

Section 2.5 explains the conceptual capabilities of frames. These include inheritance (Section 2.5.1), default values (Section 2.5.2), demons (Section 2.5.3), and perspectives (Section 2.5.4). Perspectives are explained using another frame representation language, KRL (Section 2.5.4.1). Section 2.6 describes the FDM user-view frame type, and Section 2.7, the FDM inference frame type.

2.1 BACKGROUND

In many Artificial Intelligence knowledge representation projects, databases have been implemented using frames. These databases created using frames are one variety of what are called knowledge-bases in Artificial Intelligence. Some of the databases have been designed using hypothetical application environments such as a children's story [22],

and some using commercial application environments such as an airline travel system [5]. These databases have served as test case knowledge-bases for knowledge representation projects. They have tended to be by-products of these projects and not the target goals themselves. These databases have not been designed with those characteristics which are desirable in commercial databases. However, the features of frames used in knowledge representation domains can be used to define themselves, define the database they are used to build, represent the contents of the database, and define and monitor the usage of the database.

2.2 A BRIEF HISTORY OF FRAMES

The human memory is not simply a storehouse of facts. It also maintains the means to organize and use facts. The composite of facts and the means to use facts can be called knowledge. Both Psychology and Artificial Intelligence have been interested in understanding how knowledge can be represented and what types of mechanisms might be used to activate knowledge. From this interest has stemmed the concept of knowledge representation.

It has seemed likely, for some time, that human memory makes use of some type of mechanism which allows individuals to recognize events and objects which are similar to events and objects which have been encountered before [26]. Such a mechanism would require features which allow matching, classification, generalization, and differentiation of prescriptive facts. It is clear that such a mechanism must contain, to

some degree, the semantic references intrinsic to events and objects so that these features may be facilitated.

The concept of a "frame" structure as a means of knowledge representation was first introduced to the Artificial Intelligence community by Minsky in 1974 [3]. In his paper, Minsky describes frames as a mechanism for representing stereotypical information about events and objects. "We can think of a frame as a network of nodes and relations" (pg.1). A structure of frames is generally hierarchic, with more generalized stereotypical information placed in the upper levels of the structure, and the more specific object or event information at the lower levels. Higher level frames are prototypes of related lower level frames. Within a frame, pertinent prescriptive facts may be held as well as components which link (or relate) the frame to other frames and activate the frame within the structure.

Minsky's original proposal of the frame concept was intentionally vague in terms of implementation and processing. This has caused the specific design of frames in actual systems to be based on the needs and perceptions of future designers. Since 1974 many methods of frame system implementation have been developed. Because of the range of problem domains and desired behavior of these systems, each system has somewhat different features.

A central theme to many frame systems is the ability to perform matching strategies. The field of natural language processing has seen frame systems designed and implemented for natural language understanding. Systems developed for this purpose include GUS [11] and the script system [8], which is a modification of the frame concept. The area of expert systems has created problem-solving programs using

frames. One such system is ISAAC [25] designed to solve problems in physics.

Collectively, each of these frame systems fall into the category of representational schemes called knowledge-bases [9]. Each one of these systems supports a database consisting of logical frame structures. Systems such as GUS [11] and NUDGE [6] have been developed in problem domains which are common commercial database application areas, namely travel reservations and appointment scheduling. The primary area of interest in development of GUS was the natural language interaction with the system and not the refinement of the database structure as a commercially usable product. NUDGE was developed to augment an inter-office scheduling system for appointments and meetings. Although it used a database for its processing, again the concerns of commercial database maintenance were not central to the project. Rather, the interest was primarily in the areas of recognition of information about the problem domain which must be represented in the system, experimentation with multiple representation methods, and analysis of the FRL (Frame Representation Language) as a successful implementation tool [6].

2.3 FRL TERMINOLOGY

Because FRL is one of the most easily understood languages for representing frame systems, its terminology and form have been borrowed from ref. 14 for the presentation of examples in this thesis. However, its use is for clarification purposes and is not intended to present

implementation requirements. The frame capabilities are being presented, not a method for implementation.

The basic components of the FRL frame form are shown in Figure 1. "Frame" is the frame name. The contents of the frame are subdivided and named using "slots". The two most common slots in FRL are "A-Kind-Of" (AKO) and "INSTANCE". The AKO slot acts much as an IS-A link in a semantic network [27]. An IS-A link indicates a relationship between two objects such that object A, which IS-A object B, has all of object B's generic features. In other words, A is a specification of B. An example would be that a ROBIN IS-A BIRD. The frame name(s) which appears in the datum position for an AKO slot is an immediate upper level attribute frame which acts as a prototype of the frame that contains the AKO slot. A frame name which appears in a specific attribute frame's INSTANCE slot is a member of a group of one or more frames which conforms to the specifications of the attribute frame. This concept of "conforming to specifications" is explained in the next section.

A "facet" within a slot indicates the nature of the "datum". A facet may indicate a specific value ($VALUE), a default value ($DEFAULT) when a specific value does not exist, or demon types ($IF-NEEDED) to trigger an attached procedure when a value must be acquired. Additional demons are provided to deal with situations such that some action must be performed when something is changed in the slot's datum contents, or when the frame is newly created ($IF-ADDED and $IF-REMOVED), or to define the allowable datum for a slot ($REQUIRE). The "datum" itself may be some numeric, alphanumeric, or logical constant, another frame name, or procedural statement.

```
(frame1
  (slot1 (facet1 (datum1 (label1 message1 message2 ... more Messages ...)
                          ... more Comments ...)
                 (datum2 (label1 message1 ...))
                  ... more Data ...)
         (facet2 (datum1 (label1 message1 message2 ...)))
          ... more Facets ...)
  (slot2 (facet1 (datum1 (label1 message1 ...) ...) ...) ...)
   ... more Slots ...)
```

FRL FRAME STRUCTURE

-- FIGURE 1 --

FRL also includes "labels" and "messages" (which act as comments) that can be used to clarify the contents of a frame, but which are not essential to understanding the basic form or components of a frame.

2.4 THE FRAME SYSTEM STRUCTURE

The hierarchic structuring of a frame system is defined by the semantic hierarchy of the application domain. Each level of the structure is dependent on its previous levels. The very top levels of the structure define the structure itself and the structure of its components - frames. These components are the different types of frames active in the structure. In the FDM these are the attribute, element, inference, and user-view frame types. The top levels define these frame types by defining their components and by representing their semantic uses and differences. Each frame placed below another frame, in a parent-child order, must fulfill the semantic requirements of its parent level.

To see how this structure definition works it is helpful to first look at how the structure of a "real world" example would be represented. This example might itself appear at some level in a FDM database. It is also necessary at this time to introduce two of the FDM frame types in detail.

2.4.1 THE FDM ATTRIBUTE FRAME

The first frame type which is needed in the FDM is the attribute frame. The attribute frame contains the semantic (and syntactic)

specifications to which any related lower level frame must comply. In other words, given a frame B which is linked, via its AKO slot, to frame A, then any contents of frame B must be compatible with the specifications of frame A. Bobrow and co-workers [11] describe this association by stating:

> "If one frame is the prototype of another, then we say that the second [frame] is an instance of the first [frame]. A prototype serves as a template for its instances." (pg. 163)

For example, a college student is also a person. To consider the concept of an individual being a college student is to consider all of the semantic features of being a person plus those specific features which semantically define a college student. A sample case of a frame structure which represents this relationship is shown in Figure 4 (see Chapter 2.5.1). This structure could be expanded to represent that a person is a mammal, and a college student has the semantic features, <u>via</u> the AKO slot of PERSON, of a mammal. The chain could continue to represent a mammal as an animal, and an animal as a living thing. The extent to which the chain continues up, or must be subdivided down, depends on the requirements of the domain being represented.

An attribute frame contains the stereotypical semantic definition of a group or class. It contains only the specific information which always holds true about its members. For example, an attribute frame for a mammal (Figure 2) would contain the constraints of all mammals being "warm-blooded vertebrate animals which have lungs and milk producing glands in the females" [18]. These features are common to all mammals. Any frame related at a lower level to the mammal frame must minimally be able to fit these criteria, and each shares this

```
mammal :
   AKO : $value : animal
   INSTANCE : $value : canine,person,whale,...
   SKELETAL-STRUCT : $value : vertebrate
   AIR-INTAKE : $default : lungs
   CIRC-SYSTEM : $value : warm-blooded
   ENVIRONMENT : $default : land
   .
   .
   .

whale :
   AKO : $value : mammal
   INSTANCE : $value : blue,sperm,hump-back,...
   FOOD : $default : plankton
   ENVIRONMENT : $value : salt-water
   .
   .
   .

lung :
   AKO : $value : body-organ : def : Either of two spongy,
                                     saclike thoracic organs
                                     in most vertebrates,
                                     functioning to remove
                                     carbon dioxide from the
                                     blood and provide it
                                     with oxygen
   USE : $if-needed : life-support(lung)
   .
   .
   .

moby-dick :
   AKO : $value : sperm
   LOCATION : $value : atlantic
   .
   .
   .
```

FRAME STRUCTURE ELEMENTS

-- FIGURE 2 --

information. Though the majority of mammal types live on land, it is not always true, as with whales. Therefore that information might appear as a default in the mammal frame, but not as a rule for the association of lower level frames.

An attribute frame can also hold semantic and/or syntactic constraints, as values of its own slots, which act in a template type manner to restrict the values of corresponding slots in lower level attribute or element slots. The $REQUIRES facet is used for this purpose.

2.4.2 THE FDM ELEMENT FRAME

The second type of FDM frame is the element frame. These frames should only occur at the lower levels of the frame structure. Their names are instances of prescriptive facts, such as a person's name. They contain references to their immediate attribute frame(s) in their AKO slots, and the names of other frames to which they are related. These may be other element, attribute, and inference frames, which will be described when needed. In some cases, usually in the form of a comment and for explanatory purposes, a linguistic definition will be represented. This might be a definition of the frame name or of some item in the frame. Figure 2 contains an example where the dictionary definition of "lung" is contained in the element frame named LUNG.

Many times it is necessary to represent items which have no semantic reference in and of themselves. Values of this kind are sometimes numeric (25, 4, 970), alphanumeric (bud, 9m), and logical truth values. Someone might be '25' years old, but 25 itself has no

semantic reference other than its meaning in a number system. If the database application domain is not dealing with a number system, it probably would not be useful to represent this type of semantic information. So in many cases it would not be appropriate to represent these types of items as individual frames. These items usually appear as an $VALUE, $DEFAULT, or $REQUIRE datum in either attribute or element frames. The use of $REQUIRE to handle syntactic type-checking is included in Section 4.2.

An element frame should not have an INSTANCE slot. Examples of element frames in Figure 2 are LUNG and MOBY-DICK. LUNG might be an attribute frame in some application where instances of element frames such as IRON-LUNG, NATURAL-LUNG, and ARTIFICIAL-LUNG were desirable as semantic sub-types. For this example LUNG serves its purpose as an element frame.

2.4.3 USING FRAMES TO REPRESENT FRAMES

Now that the above two frame types have been described and used to represent a real world example, it can be seen that these same frame types can be used to describe themselves. An attribute frame named FRAME would have instances of ATTRIBUTE-FRAME, ELEMENT-FRAME, INFERENCE-FRAME, and USER-VIEW-FRAME. These frames are also attribute frames and contain the semantic criteria for associated lower level frame occurrences of their type. These are very high level, or "meta-level", frames which are generally static in any particular FDM design. Changes made to these frames effect the entire database. As a

result, changes should be made rarely, with much prior consideration, and only by someone in the highest levels of authority, i.e., the DBA.

Figure 3 shows the type of information that would be represented in these attribute frames. The actual form of these frames would be dependent on the frame implementation language used, just as was stated earlier concerning the other frame examples appearing in this thesis. The use of the word reference in the figure means the occurrence of other frame names, within a given datum, in any of the frame's slots, except the AKO and INSTANCE slots.

2.5 THE CONCEPTUAL CAPABILITIES OF FRAMES

To see how an FDM can improve commercial database organization and manipulation it is necessary to explain the capabilities of frames in greater detail. There are four basic properties that frames use to share information and control activities within the frame system. These are inheritance, default values, demons, and perspectives [4].

2.5.1 INHERITANCE

Inheritance makes sharing of generic, stereotypical, and/or specific information possible. This inheritance can be vertical or horizontal among frames, depending on the requirements of the system. Vertical inheritance consists of the information contained in an upper level frame which is information also true for any lower level frame where the upper level frame appears in its AKO slot.

ATTRIBUTE FRAME

- -- contains specific information which is pertinent to its level and no higher level frame
- -- contains necessary references of higher level frames (AKO) which contain constraints which it must fulfill in all cases
- -- contains references of lower level frames (INSTANCES) which must fulfill all of the constraints which it contains
- -- may contain references of other frames
- -- contains constraints which specify its relationship to other frames which it references
- -- may contain non-frame-name items, such as numeric and string constants and boolean values

ELEMENT FRAME

- -- should not contain references of lower level frames (INSTANCES)

INFERENCE FRAME

- -- is procedurally functional
- -- may be independent of reference to other frames, except AKO at the top level

USER-VIEW FRAME

- -- may contain constraints which override other specified frame-interaction references, except of the top level

FDM FRAME TYPES
-- FIGURE 3 --

The hierarchic ordering of the frame structure lends itself to vertical inheritance. A frame which is subordinate to another frame may be an instance of that frame. In Figure 4, the frame for JONES is an instance of the frame STUDENT. Similarly, vertical inheritance upward can be indicated by specifying STUDENT to have a relation A-Kind-Of to the PERSON frame.

Horizontal inheritance, sometimes also called "indirection", is facilitated by the appearance of another frame name in the datum position of a slot. This makes information in the second frame available to the first. In Figure 4, there is a frame for CS460. A user might query the database to find out if CS460 is taught in an air-conditioned room. The datum in the ROOMS slot of CS460 is the frame name F212 and the desired information is contained in its slot FACILITIES.

Both vertical and horizontal inheritance can be used in conjunction when necessary. Again using Figure 4 as an example, a user might query whether the enrollment size of CS460 could be increased to 50 without changing the room due to capacity limits. Using horizontal inheritance the F212 frame is examined but is found to have no slot for the room's capacity. Vertical inheritance uses F212's AKO slot to examine its immediate attribute frame CLASS-ROOMS. The CLASS-ROOMS frame has a capacity slot with a default value datum of equal to or greater than 50. It is possible to respond to the user that the change should be possible without a room change being necessary.

```
jones :
   AKO : $value : student
   COURSES : $value : cs460,ee241,...
           : $if-removed : enrollment(remove,course)
           : $if-added : enrollment(add,course)
   ROOM : $value : s104,s112,f212
   ROOM-LOC : $if-needed : find-room(room)
   :

cs460 :
   AKO : $value : course
   INSTRUCTOR : $if-needed : find-inst(course)
   ROOM : $value : f212
   ROOM-LOC : $if-needed : find-room(room)
   NO-STDNT : $value : 45
   NAME-STDNT : $value : jones,...
   :

f212 :
   AKO : $value : class-room
   COURSES : $value : cs460,...
   LOCATION : $value : fairchild
   FACILITIES : $value : projector,screen,air-conditioner
   :
```

FDM ELEMENT FRAME EXAMPLES

-- FIGURE 4 --

```
student :
   AKO : $value : person
   INSTANCE : $value : jones,...
   AGE : $default : ≥17
   M-V-D : $value : course,class-rooms,teacher,...
   :

class-rooms :
   AKO : $value : rooms
   INSTANCE : $value : f212,s104,s112,...
   CAPACITY : $default : 50
   M-V-D : $value : course,teacher,student,...
   FACILITIES : $default : black-board
   :

course :
   INSTANCE : $value : cs460,ee241,...
   ENROLLMENT-SIZE : $if-needed : calc-enmt(student)
   M-V-D : $value : class-rooms,student
   BEING-HELD : $requires : room-asign(course),inst-asign
                            (course),stdnt-enroll(enroll-
                            size>15)
   :
```

FDM ATTRIBUTE FRAME EXAMPLES

-- FIGURE 4 --

(continued)

2.5.2 DEFAULT VALUES

Default values serve as acceptable values when specific values are missing. They can be used very effectively in an attribute frame when a lower level frame does not contain a needed value but a range limit value might fulfill the need for the information. Default values are represented in FRL terminology by the facet name $DEFAULT.

The previous example given for inheritance demonstrated the use of a default value when looking for the capacity of room F212. It is still not possible to definitely determine what the capacity of F212 actually is, but in the example the available default information was useful to the query.

With this example and with the example of the whale in Figure 2 it is cautioned that these default values should not be used as specifications of rules of compliance for lower level frames. These should be represented using specific values, ranges of values, and/or demons.

It is also important to carefully consider the appropriateness of allowing default values and/or limiting their use. The issues involved in these considerations are discussed in Section 5.1.

2.5.3 DEMONS

The activation of demons generally causes movement and processing within the frame structure, or grouping of frames, which is invisible to the user of the structure. Since demons act in accordance with the current state of the structure, it is not possible to predict the result of a specific instance of a demon's activities. However, it is possible

to generalize the behavior of a demon as far as its intended purpose, if well designed. Because of this it is essential that the use and design of a demon be very carefully considered before it is incorporated into the frame structure. Demons can be destructive to the contents of the structure. While this is appropriate behavior at times, demons must be designed with this potentiality in mind so that their abilities to alter the structure are completely understood by the designer, and their power limited accordingly.

Demons can be used to infer information, for horizontal inheritance, and to control the effects of value changes within a grouping of frames. Demons are able to perform their functions by activating procedural segments which are contained within the structure. Their specific participation in the FDM, and how they would be contained, is explained in Section 5.1.

Demons used to infer information in the structure should never have destructive capabilities. As will be shown further on in this thesis, the use of inference in a specific FDM design may or may not be appropriate, depending on the requirements of the user.

To again draw upon FRL terminology and form to demonstrate how an inference demon might be incorporated into a frame structure, an example appears in Figure 5 of an $IF-NEEDED facet. The occurrence of the $IF-NEEDED facet indicates that its datum value will be active in finding information which is not immediately available in the structure and must be searched for or inferred. The datum itself may be a series of procedural statements or the name of a frame which contains the needed procedural statements. In Section 5.1 a discussion of how these

```
sue :
   AKO : $value : mother,spouse,employee
   JOB : $value : doctor
   CITIZEN : $value :gb
   AGE : $value : 35
   CHILDREN : $if-needed : chain-child
              $if-removed : parent-of(remove,child)
              $if-added : parent-of(add,child)
   SEX : $if-needed : sex-type
   .
   .
   .

mother:
   AKO : $value : parent
   INSTANCE : $value : sue,...
   SEX : $value : female
   .
   .
   .

sex-type :
   AKO : $value : person-inference
   EVAL : $value : if AKO = spouse, sex not = find-spouse(sex)
                   if AKO = mother, sex = mother(sex)
                   if AKO = father, sex = father(sex)
                   .
                   .
                   .

voter-type :
   AKO : $value : person-inference
   EVAL : $value : if citizen = 'us' and age ≥ 18, vote = 'yes'
   .
   .
   .
```

FDM FRAMES WITH DEMONS

-- FIGURE 5 --

inference demons are greatly relied on for dealing with unprocessable queries will be found.

Demons used for frame structure control and maintenance are the demons which will have destructive capabilities. In FRL terminology these are represented in Figure 5 as the $IF-ADDED, $IF-DELETED, and $IF-ALTERED facets. These demons also activate procedural statements internal to the frame structure. Their major role is to maintain the consistency and integrity of the frame structure. In Figure 5 there are examples of the $IF-ADDED and $IF-DELETED facets in the INSTANCE slot of the frame MOTHER. If another frame were added with MOTHER in its AKO slot, or the frame name SUE were deleted, then the appropriate demon would be activated to make sure that this change is reflected in other areas of the structure which are affected. For example, if a frame ANN were instantiated and at some point added to the $VALUE facet datum of the INSTANCE slot of MOTHER, then the $IF-ADDED demon would make sure that MOTHER appears in the $VALUE facet datum of the AKO slot in the frame ANN.

In the FDM these demons perform the function of controlling the update and deletion anomalies which are discussed in Section 4.3.2.

2.5.4 PERSPECTIVES

Perspectives make it possible to semantically represent information from multiple viewpoints (i.e., John is a dentist, and John is a member of a bowling league). Being a dentist and being a member of a bowling team may both be true perscriptive facts about John. They each have

features which are not shared with one another. They are different "aspects" of John [17].

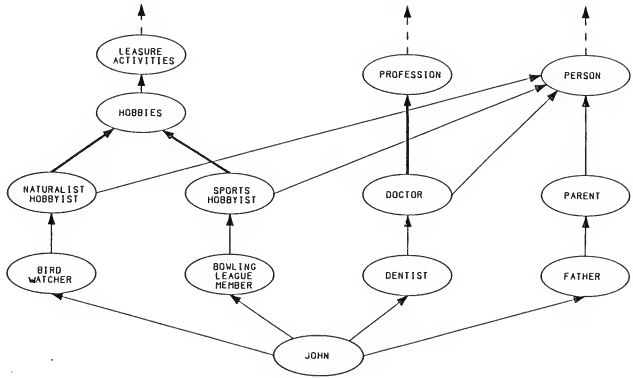
At times, the user of a frame structure may wish to locate more complete, but stereotypical, information about some individual, like John, which has a unique semantic reference external to the frame structure. There are obvious advantages to being able to do this selectively so that all of the available higher level information need not be examined, only that which is pertinent. Perspectives make it possible to examine the more likely paths to find general information from higher levels of the frame structure.

Looking at Figure 6 it can be seen that John's AKO slot would contain the frame names BIRD-WATCHER, BOWLING-LEAGUE-MEMBER, DENTIST, and FATHER. Suppose the following query were posed to the frame structure:

"John has an x-ray machine. What other types of work are done with x-ray machines?"

As stated in Section 2.4.1, only information which is stereotypical to its immediate lower level frames may be contained in that frame. For the sake of this example, doctors typically have x-ray machines. Because of this, an examination would have to be made, either using a breadth-first or depth-first search, of all of John's AKO frames and each of their AKO frames, until finally the DOCTOR frame is found.

In the best case, using a breadth-first search, it would be necessary to examine six other frames before finding the DOCTOR frame. Indeed, this is probably a contrived case. However, this type of problem can arise. Perspectives offer a semantic solution to this problem.



PERSPECTIVE TREE

-- FIGURE 6 --

2.5.4.1 PERSPECTIVES IN KRL

FRL does not have a form of representation for perspectives. They are included in KRL (Knowledge Representation Language) [5]. The previous example is the type of system processing that GUS [11], developed using KRL, was designed to handle.

In KRL the use of perspectives involves the comparison of specific item features to prototype features. When matches occur between the item and the prototype, the perspective of the prototype is available as a viewpoint with which to complete the processing.

Figure 7 uses an adapted version of the KRL representation of a perspective. The query example is the same as in the previous section. John is the specific item and the feature in question is the x-ray machine. Various samples of prototypes are given, with their stereotypical features. The features of JOHN are compared to the features of the prototypes and a match is made with the DOCTOR prototype.

The use of perspectives in the FDM to deal with types of queries similar to the example query would be advisable only if appropriate to the user's requirements. The major use of perspectives in the FDM will be in an adapted form for specifying user-view frames as described in the next section.

2.6 THE FDM USER-VIEW FRAME

User-view frames are very high level frame instances in the FDM. These frames are used to monitor and control access for their specific

```
prototype FATHER has
          feature:  children


prototype BOWLING-LEAGUE-MEMBER has
          feature:  league-meeting-time
          feature:  league-meeting-place
          feature:  bowling-ball-weight


prototype BIRD-WATCHER has
          feature:  binoculars


prototype DOCTOR has
          feature:  medical-diploma
          feature:  x-ray-machine
          feature:  nurse


prototype DENTIST has
          feature:  dental-drill
          feature:  dental-chair


item JOHN has
     feature:  x-ray-machine
```

PERSPECTIVE MATCHING EXAMPLE

-- FIGURE 7 --

area, or subscheme [2], of the FDM structure. These frames rely primarily on perspectives and demons to perform these functions.

2.6.1 SECURITY

The major difference between the user-view frame and other types of frames in the FDM is that it contains certain exceptions to the "prototype of an instance" rule set forward in Section 2.4.1. While a higher level user-view frame would still specify the semantic and syntactic constraints of its instances, its contents might <u>not</u> be available to its instances. This would facilitate security and privacy between user views.

This can be accomplished by disabling certain capabilities of its instances, such as upward vertical inheritance and demon activation. This disablement would allow the user-view frame to, in effect, lock its doors to particular types of access from instances. It is necessary that a user-view frame be able to do this, otherwise a lower level frame could change its contents, particularly the constraints it holds, which would allow uncontrolled access to unauthorized areas of the FDM structure.

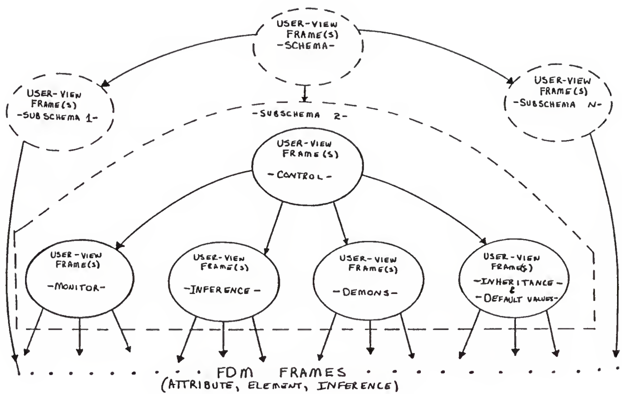
One method of enforcing disenablement would be by creating a special primitive datum type for use in $REQUIRE type facets. In this thesis this datum will be called "NONE". Again, to use the analogy of a locked door, NONE would act as a dead-bolt. NONE bars access from lower level instances to the slots in which it is contained. These slots are only accessible from higher level frames.

Depending on which slots contain this value, different types of frame access would be denied. NONE could appear in slots which make parts of the user-view frame inaccessible to the user (to peruse or make changes to the frame's contents), or to internal FDM process flow (for inheritance or demon processing for inference or frame content changes). By denying accesses of these types, system-internal intentional and unintentional tampering can be greatly reduced. This also removes much of the possibility of erroneous system processing. Generally, the use of these locked doors keeps processing, done by one user, limited to the specified user view.

In addition to making certain access and process capabilities unavailable, and thereby improving the security of the FDM, it would be easily possible to include features to the top level user-view frame(s), for the schema [21] specification of the database, to monitor which user view is attempting to break which constraints. This would facilitate the keeping of a log of the processing in the FDM.

2.6.2 THE SPECIFICATION OF USER VIEWS

User-view frames form a substructure within the FDM structure. Any one user view may be constructed of one or more user-view frames which hierarchically define the user view (Figure 8). Contained in the user-view frame(s) is the user view control capabilities. Some of these controls are used for accomplishing the security maintenance, described in the previous section, as well as to maintain consistency of accessing within its view. This can involve such thins as dealing with conflicting accessing and alterations to the database.



FDM USER-VIEW FRAME STRUCTURE

-- FIGURE 8 --

The user-view frames can override the capabilities (Sections 2.5.1, 2.5.2, and 2.5.3) of the frames associated with them. They specify any constraints on the use of inheritance, demons, and default values which lower level frames may attempt, but are inappropriate for the particular user's view of the FDM. Even though the lower level frames contain such capabilities, their activation can be stopped or limited from within the user-view frames. The use of demons can best facilitate this.

The user-view frame(s) specifies the perspective of the user's view. Unlike the KRL use of perspectives, the user view does not originate from matching an individual item to prototypes. Rather the user-view frame(s) specifies which attribute frames are included or excluded from the view. These attribute frames are those in the actual database levels of the FDM and not the metalevels.

User-view frames may also contain features for monitoring activities in the FDM while the view is active. This would provide the user with information concerning which processes have taken place. More complete information could be provided to the DBA as to how the processing was done (i.e., the paths of the processing).

The user-view frame(s) only has control of its specified areas of the FDM while it is active. The limitation or disablement of a frame feature is only effective during this time. In this way the same lower level frames may be activated for other user views and their capabilities will be able to function as is appropriate under the control of newly activated user-view frames.

Section 4.1 gives examples of how user-view frames effect the perception and use of the FDM for multiple users. These examples show

how user's views can be semantically separate, and how the user-view frames can effect user processing.

2.7 THE FDM INFERENCE FRAME

Section 2.5.3 described how demons can be used to infer information and to assure that an alteration to a frame is consistent with other frames the altered frame is associated with. Demons activate procedural statements to accomplish these functions. It is often the case that the same procedural statements need to be activated from many different frames within a frame structure.

If a user of a frame structure had to repeatedly place the needed procedural statements in these frames as the datum values for demon facets, it would only be a matter of time before some error occurred in the copying. In addition, if a specific group of procedural statements needed to be altered very frequently and there were many occurrences of the group, more processing would be required to change all occurrences than if only one occurrence existed.

In FDM an inference frame (so named even though they are for all demon functions) would contain the only occurrence of a demon's needed procedural statements. The demon facet's datum would be the needed inference frame's name. In this way the inference frame is mutually accessible by the frames which must make use of it.

Aside from the obvious advantages of modularity, the use of inference frames also facilitates greater consistency and control of demon activity. There is only one version of a specific set of procedural statements, thus any erroneous behavior can be easily traced.

Inference frames also make it easier for the user-view frames to control demon activity. This is done by disabling access to the inference frame itself instead of finding and disabling each demon facet which uses those procedural statements.

Section 5.1 demonstrates some of the types of procedural statements which might be contained in inference frames and gives examples of their use in the FDM.

CHAPTER 3
ORGANIZATIONAL ISSUES

Chapter 3 deals with the issue of representing multivalued dependencies (one-to-many and many-to-many relationships) in database models. (Given a set of values, hereinafter referred to as an attribute, a multivalued dependency will occur when one or more values of a specific attribute functionally determines more than one value of another attribute.) When one value of the domain attribute functionally determines many values in the range attribute, a one-to-many relationship exists between these attributes. When many values of the domain attribute functionally determine many values in the range attribute, a many-to-many relationship exists.

Section 3.1 offers examples of the types of semantic references that multivalued dependencies may have in the real world. Section 3.1.1 discusses how the network, hierarchic, and relational database models deal with multivalued dependencies. Special attention is given to the relational model's fourth normal form. Section 3.2 discusses how the FDM can deal with multivalued dependencies.

3.1 MULTIVALUED DEPENDENCIES

A very large number of the facts to be represented in databases are members of attributes which have multivalued correlations to other attributes. These can be called multivalued dependencies and may be one-to-many or many-to-many relationships. A one-to-many relationship is illustrated by an individual parent with multiple children, a person who is multilingual, or by a salesperson who works in multiple

sales-districts. A many-to-many relationship might be illustrated by many individuals multilingual in the same languages, or by salespeople who work in more than one district but are only one of many who work in a specific district.

Because of the amount of information which is multivalued, databases need to be able to support multivalued dependency representations. A database which is not able to do so will not be able to be a natural representation of its environment. In the standard database models, multivalued dependency relationships are dealt with using methods depending on which specific model is being used for representation.

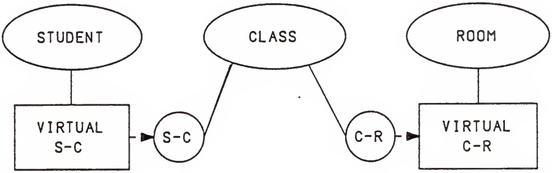
3.1.1 MULTIVALUED DEPENDENCIES IN THE NETWORK, HIERARCHIC, AND RELATIONAL MODELS

The network model could, conceptually, allow both types of multivalued dependency at the intentional modeling level. However, in order to provide a network which is a directed graph, a many-to-many multivalued dependency is transformed into two one-to-many multivalued dependencies (Figure 9). The directed graph is a simpler structure to conceptualize and this, in turn, makes the extensional modeling task easier.

The hierarchic model restricts the occurrence of multivalued dependencies to the one-to-many relationship because of the basic structure of hierarchic ordering of nodes, as well as for the ease of extensional modeling. In order to create a hierarchically ordered model it is necessary that there be a parent-child relationship between nodes. For this reason, many-to-many relationships are again transformed into

Network Model

Directed Graph Network Model



Hierarchic Model

N-M AND 1-N DEPENDENCIES IN
THE NETWORK AND HIERARCHIC MODELS

-- FIGURE 9 --

pairs of one-to-many relationships and virtual records are used as pointers at the extensional modeling level (Figure 9).

The occurrences of multivalued dependencies are easily dealt with in the network and hierarchic models and do not cause any functional problems at the intentional level, though the limitation of using only one-to-many multivalued dependencies makes a completely natural representation of the application environment impossible. The relational model is a completely different case from the network and hierarchic models and must be looked at in greater depth.

The relational model allows only one-to-one relationships in the first three forms of normalization. The fourth normal form allows for one-to-many relationships through a dependency called an MVD. An MVD, X ->-> Y, exists in a relation R (X,Y,Z) if, and only if, Z is independent of, i.e., has no relationship to, Y. Then X ->-> Z also exists [19]. The practical value of fourth normal form is questionable when querying those relations which contains an MVD.

Forth normal form restricts the number of multivalued dependencies occurring within one relation to one. If there were three attributes which were related to each other as multivalued dependencies they would have to be broken up into two separate relations. For example, the attributes STUDENT, COURSE, ROOM are often many-to-many relations. Each student may have many courses, those courses may be held in different rooms at different times and generally will have more than one student. Students will often have courses in different rooms if more than one class-room exists and each room will likely have more than one student in it, and be used to teach more than one course. The relationship of courses to rooms, in addition to the relationships of students to

courses and students to rooms, also disallows this set of attributes to be in a fourth normal form relation.

As a sample case, there is a student (JONES) who is taking two courses CS460 and EE241, where course CS460 is taught in room F212 and course EE241 is taught in rooms S104 and S112. Figure 10 is a representation of the sample case in an unnormalized form. Figure 10.1 is a representation in fourth normal form where the relation has been broken up into two separate relations.

In order to directly access any tuple in either of the multivalued dependency relations, it is necessary to have each of the multivalued attributes contained in the key. This makes the fourth normal form impractical since to directly access any tuple for querying, all of the values of that tuple must already be known. Querying a fourth normal form relation can only be useful in instances where the user wishes to verify that a given tuple does currently exist in the database.

It is also a problem that fourth normal form requires there to be multiple relations to represent multiple one-to-many multivalued dependencies. In the sample case, the relationship between courses and rooms is no longer represented. It is not possible to query the relations in Figure 10.1 to find out the room in which CS460 is taught. The semantic correlation of courses and rooms cannot be represented unless an additional relation (Figure 10.1a) is placed in the database. All of the information in Figure 10.1a is redundant. Since this is a simple case given as an example, the amount of redundancy is not great. As the number of inter-related multivalued dependency attributes increases, the redundancy increases proportionally.

STUDENT	COURSE	ROOM
JONES	CD460	F212
	EE241	S104
		S112

UNNORMALIZED RELATION EXAMPLE
-- FIGURE 10 --

STUDENT	COURSE
JONES	CS460
JONES	EE241

STUDENT	ROOM
JONES	F212
JONES	S104
JONES	S112

FOURTH NORMAL FORM OF FIGURE 10
-- FIGURE 10.1 --

COURSE	ROOM
CS460	F212
EE241	S104
EE241	S112

ADDITIONAL RELATION TO FIGURE 10.1
-- FIGURE 10.1a --

A violation of fourth normal form, such that more than one one-to-many multivalued dependencies are allowed in a relation, requires the selection and use of an alternative organizational method for the relation's contents. The methods discussed in this thesis were presented in ref. 1, and are all examples of two one-to-many multivalued dependencies represented in one relation.

The method of representing the contents of a multiple multivalued dependency relation using a cross product of attribute values (Figure 5.2) results in massive amounts of redundancy of information, may cause update and deletion anomalies, and still requires inclusion of all multivalued dependency attribute values in the access key. (Specific issues dealing with update and deletion anomalies are discussed in Section 4.3.) The use of a cross product representation may also cause erroneous information to be contained in the relation. This is unavoidable, since each possible combination of attribute values must appear in the relation, whether or not the combinations actually exist. For example, Jones' course CS460 is actually held in room F212 and not in any other room. Because of the cross product results, tuples in which course CS460 is associated with rooms S112 and S104 must also be represented. In the case of Figure 5.2 fifty percent of the tuples in the relation are erroneous.

Figures 11.2 and 11.3 present two methods of representation, each using null values. In both cases the likelihood of update and deletion anomalies is high. In Figure 11.2 the correlation of courses to room numbers has again been lost so that it is not possible to reliably query the room in which a specific course is taught, or which courses are taught in a specific room. Figure 11.3 shows most of the correlations

STUDENT	COURSE	ROOM
JONES	cs460	F212
JONES	cs460	s104
JONES	cs460	s112
JONES	EE241	F212
JONES	EE241	s104
JONES	EE241	s112

CROSS PRODUCT OF FIGURE 10

-- FIGURE 11.1 --

STUDENT	COURSE	ROOM
JONES	cs460	
JONES	EE241	
JONES		F212
JONES		s104
JONES		s112

DISJOINT REPRESENTATION WITH NULL VALUES
OF FIGURE 10

-- FIGURE 11.2 --

STUDENT	COURSE	ROOM
JONES	cs460	F212
JONES	EE241	s104
JONES		s112

MINIMAL NUMBER OF VALUES WITH NULL VALUES
OF FIGURE 10

-- FIGURE 11.3 --

of courses to rooms, but it is still not possible to tell which course is taught in room S112.

The representation of the sample case values in Figure 11.4 is a major improvement in terms of performing reliable queries to the relation. All of the information is available, and in correlated form, for all attributes of the multivalued dependencies. However, this representation is still prone to maintenance anomalies.

The examples of the two minimal-number-of-values forms represented in Figures 11.3 and 11.4 are not random mixes, as these two forms are in Kent's examples [1]. Problems with erroneous tuples would arise in both cases if the relation were randomly mixed. Since representations with non-random mixes are already error prone, there seems to be little need to include random mixing. It is not difficult to conceptualize the maintenance and query errors which would arise from randomly mixing the values in the minimal-number-of-values method. Specifically, the correlation between courses and rooms in Figure 11.4 would be lost.

Because of the extreme nature of Kent's unrestricted-random-mix method, an example of the representation of the sample case using this method has not been included in the figures or discussion.

Given the mechanical problems with representations of multiple multivalued dependency within one relation, it is necessary to conform to the non-violated fourth normal form. However, as pointed out above, not only is query processing of minimal use, but only two of the semantic correlations of the sample case attributes can be represented, STUDENT to COURSE and STUDENT to ROOM, without the addition of relations which contain completely redundant information (Figure 10.1a).

STUDENT	COURSE	ROOM
JONES	CS460	F212
JONES	EE241	S104
JONES	EE241	S112

MINIMAL NUMBER OF VALUES WITH REPETITIONS
OF FIGURE 10

-- FIGURE 11.4 --

3.2 MULTIVALUED DEPENDENCIES IN THE FDM

In an FDM the representation of either one-to-many or many-to-many relationships are not restricted in any way. Both types of multivalued dependency can be directly used in the model with no difficulty. The FDM does not deal with sets of values, as with an attribute, but rather deals with individual element frames which represent each prescriptive fact. These element frames can be linked to others, for multivalued dependency purposes, by containing multivalued dependency and functional dependency (one-to-one) slots.

The problem of needing to know all the values within a group of values in order to access those values, the major problem with the non-violated fourth normal form, does not arise either. The FDM can use the links within an element frame to access its associated element frames.

The attribute frame can include a specification as to whether its lower level element frames may be multiply related to the element frames of other attribute frames. In Figure 6 the attribute frames COURSE and STUDENT contain each other in their multivalued dependency slots. Since the frames are reciprocal, they have a many-to-many relationship. In order to show how a one-to-many multivalued dependency can be specified, the multivalued dependency for COURSE does not contain the attribute frame name TEACHER. (It is often the case that one course will be jointly taught by two or more instructors, but for this example a limit of one instructor per course will be imposed.) If the attribute frame TEACHER has a multivalued dependency value of the attribute name COURSE, then there is a one-to-many relationship from TEACHER to COURSE.

An individual FDM design may have a default of one-to-one multivalued dependencies if no attribute name is specified, and may or may not have its name appear in the multivalued dependency slot of another attribute frame. It may be judged, by the particular system's designer, to be more appropriate to have a one-to-one slot specified to contain the correspondent attribute frame name in the case of a one-to-one relationship. The inclusion of both functional and multivalued dependency slots would comprehensively specify the ways in which an attribute frame and its lower level element frames could be related to others in the database.

CHAPTER 4
ACCESS ISSUES

4.1 MULTIPLE USER VIEWS

Databases are most cost-efficient when shared by multiple users. Each user should have access to that portion of the database which is pertinent to their application area, but to no more. In an FDM, frame instances can be specified to define a given user's view. Such a user-view frame would monitor and control the activities which that user may perform while interacting with the database.

User-view frames would make it possible for the user view to be a completely individual perspective of the database. These frames can direct the semantic reference of the user view. In this way the manipulation of the database can mirror the reference perspective of the user's application. A specific user view would not only differ from another user view by access rights and difference of access paths, but would also provide a semantically specific view of the database.

For example, a database might contain an inventory for a small grocery store. This inventory may need to be analyzed by different users of the database for different reasons. An accountant might examine the inventory to produce a quarterly tax report and is concerned with which items are currently in the store. A police officer, on the other hand, may be investigating a robbery in the store and is looking for missing items, possibly by comparing the old inventory to a new inventory taken after the robbery. A customer may examine the inventory to see not only if a given item is sold in the store, but whether a specific brand or size of that item is sold, as well.

The user-view frames can also control the processes which are allowed to be performed which change the contents of the database. A standard example is the case where one user may have access authority only to retrieve information and another user may have authority to retrieve and alter the same information. It is desirable to incluce mechanisms within the schema user-view frames which can assure that the access authority of any one user is enforced.

The user-view frame would limit the activation of inference mechanisms in the database during that user's access by masking or enabling specific inference frames. This is important because some application areas are flexible in the degree to which retrieved information may be non-factual. For example, a company may wish to access a database to get a listing of potential customers to contact. If this user were interested in persons in a general salary range with few current financial obligations (no mortgage payments, one or two children, etc.), the use of inferred and default values in accessed element frames would yield a more useful query result than if only factual values were used in the query.

On the other hand, other users may require only factual information to be the results of queries for their application areas. An example would be database access for tax or payroll purposes. In such a case, the user-view frame would limit the behavior of the database so that no inferred or default values are available.

These user-view frames could be used to limit and monitor the abilities of the user to make actual alterations in the contents of the database. This can be especially important when multiple users are accessing the database at the same time. Unless strict monitoring and

control is exercised over the alteration of the database contents, the alterations made by one user can make information unavailable or erroneous to another user. It is necessary that the database monitor the different active user-views, so that cross-checking of user needs can be done and used to minimize conflicts. This requires more processing time but can provide greater assurance of compatibility across multiple users' needs.

For example, if one user needs to delete items from the database (whether the item is represented in an element, inference, or attribute frame), the deletion of those items may affect another user who still needs them. The database can remove the links to those items for the first user's view, but may be designed not to remove the item itself, so that it is still available to the second user's view.

4.2 REPRESENTATION OF INCOMPLETE INFORMATION

The representation of incomplete information is a problem which occurs in the relational database model. This problem is also referred to as the insertion anomaly [2]. The occurrence of this problem arises when a tuple is to be added to a relation but not all of the tuple elements have a value. If the empty elements have null values placed in them, how will it be possible to later replace the null values with specific values? If the null value is placed in an element whose attribute is a key to the relation, will it be possible to access that tuple in the future?

Because of the difficulties associated with using null values to fill in empty tuple elements, the general method of dealing with

incomplete tuples is to not allow them. However, the values of the tuple which do exist may be needed in the database. They are pieces of information which belong in the database.

The second normal form of relational normalization was largely developed to eliminate insertion anomalies. The representation of incomplete information is a problem which does not arise in the FDM.

Any given instance of a frame contains only that information necessary to that instance, and to the level of that instance, within the FDM. An element frame instance of a specific data element (a value token of an attribute) should never exist unless at least one higher level frame exists which represents the semantic reference of the attribute. The higher level frame(s) must also contain the syntactic specifications of a data element instance of the attribute. In this way it is possible to specify the semantics and syntax of any current or expected data elements of that attribute.

Those frames which contain the semantic and syntactic constraints of an attribute can be said to specify the class membership of the attribute. Any element frame instance which is added to the database must fit the specifications of its attribute frame(s). If the new element frame instance does not conform to the attribute frame(s) the addition of the item is disallowed and the user is informed of the conflict.

For example, if a user attempted to add the element frame instance of DOCTOR to the database under the attribute of ADDRESS-CITY, the entry would not be allowed. The syntax of the new value may be valid (a character string of appropriate length) but the semantic reference of the attribute frame for cities would not allow a value such as DOCTOR.

Obviously, some types of data element values would still be error-prone. For example, it would be very difficult to check a numeric value of the correct syntactic specification for semantic specification of an attribute frame such as OCCUPATION-SALARY or HOME-MONTHLY-PAYMENT. A value such as 700.00 could be valid for either attribute, but not correct for both. By using attribute frames, the validity-checking for element frame instances would be improved, but not guaranteed.

Other than the problem of maintenance anomalies which arise in the relational model when incomplete information is allowed, there is the argument that certain information is so closely linked in the real world that its omission is erroneous. That all persons have a sex, and only one sex, is such a linked information set. In many cases it is possible to find a value for a missing data element, when its second association is this closely linked, by using inference, inheritance, and default mechanisms. In a relational database, if a relation existed with attributes including SEX and NAME, the addition of a tuple containing data elements for each attribute except SEX would not normally be allowed.

In the FDM, it is possible to reliably infer a value for an individual's sex depending on which other generic frames exist and which element frames associated with the individual are currently available at the time of query. The database can infer the sex of an individual if other information is present in the database such as the sex of the spouse, if married, whether the individual is a mother or father, if a parent, or whether the individual is a brother or sister, if a sibling. More discussion of the processing of queries when incomplete information exists will be given in Section 5.1.

4.3 UPDATE AND DELETION ANOMALIES

Anomalies in databases can be caused when the contents of the database are erroneously altered. These anomalies take the form of information being lost or inaccessible and information being inconsistent. The former case is known as a deletion anomaly and the latter is known as an update anomaly [2].

Because of the limited knowledge a user can have of the actual organization and current contents of a database, it is unreasonable to expect the user to be able to safeguard against these anomalies when performing maintenance processing. As a result, each of the standard models includes features which are intended to eliminate the risks of anomalies occurring.

4.3.1 ANOMALIES IN THE NETWORK, HIERARCHIC, AND RELATIONAL MODELS

In the network and hierarchic models, virtual records are used at the extensional level to eliminate these anomalies [2] as well as the insertion anomaly. (See Section 4.2 for discussion of insertion anomalies.) Update and deletion anomalies pose a more difficult problem in the relational model. In fact, the development of the second and third normal forms [13] was primarily motivated by the need to eliminate insertion/deletion and update anomalies, respectively.

There are two major reasons why these anomalies are so difficult to control in the relational model. First, the prescriptive facts represented in a relation which has more than one attribute are grouped together in the tuples of that relation. Therefore, an alteration of one fact, or element, of a tuple effects an alteration on the entire

tuple. Second, when redundancy of a fact occurs within the database, an alteration of that fact within one tuple may cause the redundant occurrences of that fact to be inconsistent.

When the alteration of a fact is its deletion, a void is created in the tuple in which the fact was contained. Unless the database is designed to facilitate null values as place-keepers for deleted facts, the remainder of the affected tuple is also deleted. (The use of nulls as place-keepers also creates many problems. For example, if there are multiple occurrences of nulls, how will they be differentiated in later updating? Also, if the deleted fact is a key, or part of a key, how will the effected tuple continue to be accessed?) In such cases, and when the remaining facts within the tuple are still needed in the database, the loss of the tuple causes the loss of the integrity of the database.

For example, Figure 12 shows that if all the courses taught by Smith were deleted from INSTRUCTORS, then Smith and the rest of the facts regarding Smith are lost even though Smith may still be an instructor and have an office and phone. In the case of Figure 13, if Smith were intentionally deleted from the FACULTY relation, possibly because he is no longer employed as an instructor, it must be possible to be certain that this is consistently reflected in COURSE. Either the courses would no longer be offered, and therefore removed from COURSE, or a new instructor would be assigned Smith's courses. If the courses are canceled, then CLASS-STATS must reflect this. If a new instructor were assigned to Smith's courses, then this is an update to COURSE. This change must also be reflected in FACULTY to insure consistency.

NAME	OFFICE	PHONE	COURSE	ROOM
MILLS	214	4855	405	212
BROWN	60	4498	798	212
SMITH	219	4450	305	212
MILLS	214	4855	305	208
MILLS	214	4855	720	202
SMITH	219	4450	460	208
SMITH	219	4450	420	212
BROWN	60	4498	300	202
JONES	144	4710	720	212
MILLS	214	4855	630	208

INSTRUCTORS (NAME,OFFICE,PHONE,<u>COURSE,ROOM</u>)

INSTRUCTORS RELATION EXAMPLE

-- FIGURE 12 --

NAME	POSSITION	OFFICE
MILLS	INST	214
WHITE	RSRCH	128
BROWN	INST	60
SMITH	INST	219
STEIN	CHAIR	105
JONES	INST	114
AMES	RSRCH	204

FACULTY (NAME,POSITION,OFFICE)

NO	INST	TIME
798	BROWN	T 11:00
305	MILLS	W 1:30
720	JONES	W 9:30
300	BROWN	T 1:05
460	SMITH	T 9:05
305	SMITH	W 8:30
420	SMITH	W 3:30

COURSE (NO,INST,TIME)

NO	OFFERED	ENROLLMENT	ROOM
720	SPRING	35	212
300	FALL	200	202
460	SPRING	50	208
305	FALL	100	208
798	SPRING	15	212
305	SPRING	100	212
420	FALL	50	212

COURSE-STATS (NO,OFFERED,ENROLLMENT,ROOM)

FACULTY, COURSE, COURSE-STATS RELATION EXAMPLES

-- FIGURE 13 --

This type of alteration can cause update anomalies. When a fact is changed, it must be reflected appropriately throughout the database.

Another example of an alteration which can lead to an update anomaly would be if Smith were to change his position to research. For this example it is assumed that research faculty do not also teach courses. The change to FACULTY would have to cause a change in COURSE, namely a deletion of the COURSE tuples in which Smith is the instructor. As a result either facts in the database might be lost or become inconsistent.

Depending on the maintenance procedures applied to the database for update and deletion, there may be some way to inform the user that an anomaly is likely to occur, possibly even what type and where. However, it may not be possible to guarantee that all areas which might be affected could be identified.

Through the the use of various levels of normalization the probability of anomalies can be greatly reduced. Second normal form decreases insertion and deletion anomalies and third normal form and Boyce-Codd normal form greatly reduce update anomalies. Boyce-Codd imposes the strongest restrictions in order to eliminate update anomalies for functional dependencies, however, because of the decomposition of relations. Some of the functional dependencies (one-to-one relations) of the original relations may be lost during the decomposition [2].

4.3.2 UPDATE AND DELETION ACTIVITIES IN THE FDM

The FDM eliminates anomalies by means of monitoring the alteration activities a user attempts to perform on the database contents. In situations where these activities affect other portions of the database, the FDM itself can invoke procedures which insure the consistency after the alteration is made. The traditional design and use of frame systems makes the avoidance of anomalies a standard feature. In most frame systems, the lower level frames (the element frames in the FDM) are usually dynamic in the structure. Because of this the frequent deletion and addition of element frames in FDM can be handled by standard frame system methods.

Since the general structure of the FDM is hierarchic, as discussed in Section 2.4, the FDM is able to take advantage of the organizational and pointer linkage features that the hierarchic database model uses for anomaly control. Where the hierarchic model is able to independently alter or delete the value of a single data field, the FDM is able to alter a specific fact independently of other facts in itself. There is only one instance of any fact, which is represented as the name of an element frame. Where the hierarchic model uses virtual records as a means to associate, or link, data, the FDM is also able to link facts to one another. The name of any frame is never changed, it is only deleted, thereby deleting the frame instance. The change of a fact is done by replacing a deleted frame with a new frame instance.

Demons, such as $IF-ADDED and $IF-DELETED in the FRL system [14], are used to control the effect of a deleted or added frame on other frames in the database. These demons can add or delete the frame name in the instance slot of upper level frames. If the deleted element

frame name has a link to another element frame for purposes of lateral inheritance or appears in a MVD or FD slot, the demons can insure that the deletion is consistently reflected in the linked frames. If a deleted element frame is a key in the accessing of other element frames, the demons can trigger attached procedures. Such procedures could create alternative access routes, if necessary, or inform the user of possible error hazards. It may also be the case that if another element frame is only accessible by the deleted fact, other than by its attribute frame(s), it may be appropriate for that element frame to be deleted as well.

CHAPTER 5
QUERY ISSUES

5.1 UNPROCESSABLE QUERIES

Consider a user of a database who wants to get information about persons who might be worth canvassing for a political candidate. The query "Is Sue a registered voter?" is submitted to a database. Figure 5 shows all of the data about Sue that is contained in the database. A database with no inference capability would return either a result of "NO", or that no value for registered voter status exists.

In the FDM, another special frame type can be instantiated which can be used for specific inferences. Inference frames contain semantic rules which are known about query and/or attribute domains. In this case, an inference frame could be created which holds rules concerning the criterion of being a registered voter. One of the rules that could be placed in this inference frame might be that a person must be a United States citizen and eighteen years of age or older in order to register to vote.

If the same query were applied to the database after this addition, the database would first find that no specific registered voter status value exists for Sue. The lack of a voter status result would cause activation of associated inference frames. The inference frame for being able to register to vote would reference the data that is available about Sue to find out if she fits the necessary criteria. The result of the query would be that Sue would not be able to register because she is not a United States citizen. In this case it is possible to return a definitive response and a reason for the response.

If the value of CITIZEN for Sue were United States, it would not be possible to return a response concerning her voter registration status but it would be possible to respond that she is eligible to register.

These inference abilities provide a wider range of usable information based on the specific data that is contained in the database. (It is also possible, through inference, to establish that none of the contents of the database will satisfy a given query, thereby avoiding pointless further processing.) The degree of reliance placed upon the use of inference frames (and/or default values for processing queries when specific data values are not present) depends on the requirements of the particular user of the database. When the responses to queries are to be used in areas where a high level of certainty is not needed, such as political canvassing and identification of potential customers, inferred and default responses may be appropriate. Indeed, such responses may be more helpful to the user. Were the user's requirements to necessitate responses with a high level of certainty (statistical analysis and legal uses), it may be more appropriate to recommend allowing little, if any, use of inference and default mechanisms.

The inference and default mechanisms should always be present for potential use in the database, but the degree to which they are to be active must be specified in the instantiation of a specific user's view frame.

5.2 QUERY OPTIMIZATION USING SEMANTIC CONSTRAINTS

Semantic query optimization is the technique of improving query processing efficiency by taking advantage of the semantic constraints in the query domain. A semantic constraint is a limiting feature that is known about a specific piece of information. For example, Figure 14 contains a list of items and poses a question about those items. The question is, "Which of these items can be placed in a shoe-box?" The answers are, of course, the frog, pin, coffee cup, and the dice.

In most cases a conscious consideration of the limiting features of these items, such as size or mobility, is not necessary to answer correctly. However, if consideration is given to the features of a shoe-box and the items listed, certain aspects of limitation emerge which can be made into general rules or constraints that can be applied to any list of items in the answering of the shoe-box question. A shoe-box can only hold items whose size is less than the size of the shoe-box. This eliminates the automobile, person, elephant, and Nebraska from consideration. There is not enough information available about the shoe to know if it will fit or not. The shoe could be a boot and the box could be for sandals.

Shoe-boxes can only hold those things which can be contained by cardboard. This constraint eliminates the screem (and Nebraska also, though the size rule has already done so) from items which can be given in response. An interesting point to note about Nebraska is that the reader would not usually consider the word "Nebraska", but rather thinks of the geographic state with the proper name Nebraska. The name Nebraska is a member of a group, or class, of names which all refer to different states in the United States. The decision to include or

Which of these items can be put in a shoe-box?

-- an automobile

-- a frog

-- a person

-- an elephant

-- a pin

-- a scream

-- a coffee cup

-- a shoe

-- Nebraska

-- dice

SEMANTIC CONSTRAINT EXAMPLE

-- FIGURE 14 --

exclude Nebraska from the correct answer to the shoe-box question is most likely made based on the common-sense knowledge that it is not possible for any state, even Rhode Island, to be put in a shoe-box.

So it is convenient and effective to limit the domain of eligible responses when answering any question. It saves time and energy. This way it is not necessary to actually attempt to put each item in the shoe-box to see if it will fit.

Similarly, it can be effective and cost saving to apply these types of semantic constraints to a database query, and thereby reduce exhaustive searches.

5.2.1 TWO METHODS OF SEMANTIC QUERY OPTIMIZATION

Outside of the field of Artificial Intelligence, not a great deal of research has been done, at this time, to design methods of query optimization using semantic constraints. The two major methodologies which have been developed in this area are QUIST (QUery Improvement through Semantic Transformations) [7,16,23] and KBQP (Knowledge-Based Query Processing) [15].

Both QUIST and KBQP are designed for application to relational model databases, though KBQP is also intended to be used with the SDM (Semantic Data Model) [21]. Both systems preprocess queries put to the database to transform them to equivalent, but more cost efficient, query forms. Both systems are capable of limited inference. Further both systems use the semantics of the database application domain to create sets of semantic constraints to be used in transforming the queries.

QUIST accomplishes these transformations by first looking for "constraint targets" in the original query. These constraint targets are relations which contain attributes to which semantic constraints may be applied, primarily for domain restriction. It then applies a set of constraint rules and decision heuristics to the constraint targets to find potential equivalent query transformations which will lower processing costs.

KBQP uses "techniques" (i.e., domain refinement and mapping substitutions) and "methods" (i.e., looking in the knowledge-base for equivalent facts which show that a smaller domain is equivalent to the original query's domain) to perform these transformations. When methods such as this result in positive deductions of equivalency, the results are added to the knowledge-base, as a new fact, so that it need not be deduced again in the future.

Two primitive types of expressions are used in the KBQP techniques, restriction (which acts as Codd's THETA-SELECT [24]) and image (which acts like Codd's PROJECTION). A control structure, which includes a scheduler, oversees the knowledge-base and the activities of the techniques and methods. The scheduler uses internal hueristic rules to set priorities for processing potential query transformations based on likelihood of their success.

5.2.2 SEMANTIC QUERY OPTIMIZATION IN THE FDM

The methodologies of either QUIST or KBQP can be incorporated into the FDM. In fact, the tools necessary to implement either methodology already exists in the FDM design and would naturally facilitate

incorporation and use. The "methods" and "techniques" of KBQI can be represented using inference frames. The positive deduction results, from use of the methods, can be represented as element or attribute frames. The constraint rules of QUIST can also be represented in element or attribute frames. "Constraint targets", rather than being relations which contain constrainable attributes, would be attribute frames themselves. Inference frames would examine the query for constraint targets by examining the appropriate attribute frame to determine if it has instances of other attribute frames which can refine the domain of the original query, but still produce an equivalent query.

The greatest advantage to incorporating one of these methodologies into the FDM is the existence of the user-view frames. The user-view frames are already control structures for the FDM, at both the schema and subschema levels. None of the discussions of either of these methodologies mention the ability to deal with semanticly different multiple user views. Nor can it be derived from the discussions how this might be accomplished. Would it be necessary to have separate and independent versions of implementations? The user-view frame substructure for any user view can contain the processing control, evaluation hueristics, and semantic constraint specifications for query optimization specific to its needs.

CHAPTER 6
THE SEMANTIC DATA MODEL

It would not be possible to conclude this thesis without some discussion of the Semantic Data Model (SDM) [21]. The SDM was designed as a means of "capturing" the semantic references of an application environment in its database schema. By being able to capture these semantic references, the database is a truer reflection of its application environment, and the processes which are applied to the database are more semantically meaningful.

It is my contention that the SDM does not go far enough to effectively accomplish this goal. An SDM database does not actually include the semantic references of the application environment, but rather is designed using them as guidelines for its structure, organization, and access. In addition, while one of the suggested uses for an SDM is to act as documentation for users of an implemented database (which might be the extensional level representation of a standard model type or an SDM), the SDM used for extensional modeling would not provide the "user friendliness" it could and should.

In Section 6.1 the components of the SDM are described. Section 6.2 is a criticism of the SDM.

6.1 COMPONENTS OF THE SDM

The SDM is built of entities, classes, interclass connections, attributes, and a set of primitives. The entities of the SDM represent the "objects" of the application environment. The classes are groupings of entities which are representative of the groupings of objects in the

application environment (the example given involves ship-names, countries, captains, and ship inspections). Interclass connections are used to specify the semantic relationships and hierarchic structure among base classes and non-base classes. A base class is a top level class which is not a subclass of any other class (i.e., ships), whereas a non-base class is a subclass of at most one base class and/or other non-base classes (i.e., merchant ships and oil tankers are subclasses of ships).

There are two basic types of attributes in the SDM, class and member. Class attributes are features of an entire class. Member attributes are features which are common to all members (entities) of a class. Primitives are used to derive attributes from within the database, and to define interclass connections.

The components of the SDM make it possible to support limited inheritance and inference. Multivalued relationships are also allowed to occur between classes and their members. Null values are allowed to occur as values for non-mandatory attributes and are treated as "don't cares".

6.2 A CRITICISM OF THE SDM

As stated before, the major drawback to the SDM is that it does not sufficiently represent the semantic references of the application domain. These semantics are used for defining an SDM database, but they are not contained within the database. The classes, interclass connections, and attributes are specified using the application environment semantics, and the application semantics must be known by

the users of the database for access and processing. This need on the part of the users should be minimized, and can be facilitated by the semantics being represented in the database. For example, in ref. 21, oil tankers and merchant ships are both subclasses of ships. This is a true semantic relationship among these groups. However, there is no representation of "to be a subclass" contained in the database.

In Appendix B of ref. 21, a suggested data definition language, presented in a stylized BNF, is given for the specification of a SDM. This DDL includes specifications for all the components of the SDM, from SCHEMA <- <<CLASS>> to NUMBER OPERATOR <- [+;-;*;/;!]. In the FDM this specification would itself be contained within the database. The representation would be at the "metalevel" described in Section 2.4.3. The representation of the SDM specifications is not mentioned in the presentation of SDM. It is necessary, for a model to be a semantic representation of an application environment, for the model to contain the representation of the semantics themselves.

The result of omitting the representation of the semantics within the SDM results in four areas of limitation in using the SDM. First, an implementation of a SDM database will be semantically static. Second, the representation of semantically different multiple user views is not possible. Third, a database so designed is not able to monitor and control itself using the semantic references of the application environment. Finally, a user of the database must be aware of the semantic references used in its design to be able to effectively interact with the database.

The SDM does support multiple user views to a limited extent. The user views are completely independent of the database. The SDM supplies

the ability to access the same portions of the database content using different access routes. This is done by interclass connections derived using predefined inversion and matching specifications among class membership attributes. For example, the attribute Ships-registered-here of the class COUNTRIES is specified as the inverse of the attribute Country-of-registry of the class SHIPS, and vice versa. It appears that only one inverse specification is allowed for any one member attribute. It is likely that the restriction of one inverse specification is due to the lack of a semantic representation of the "meaning" of the inverse relationships between two member attributes. Without this, it is not possible to differentiate between multiple inverse specifications from within the database. It would be necessary that the users be aware of each inverse specification and be responsible for their appropriate use.

The capabilities and components of the SDM can all be included in the FDM without including its limitations. Many of the concepts presented in the SDM design would be useful to include as access-processing strategies. Unfortunately, the limitations of the SDM cause me to conclude that it is better suited to being used for a documentation aid than as an extensional database model.

CONCLUSION

This thesis has introduced a frame based database model called the Frame Database Model (FDM) as an alternative database model. The point has been made that the databases of the future must not only be reliable and consistent, but also must be natural and unrestricted representations of their application environments. In order to achieve these two goals, the FDM is designed to contain the semantic references of the application environment as well as the prescriptive facts which must be represented in any database.

Frames have been presented as the conceptual modeling tools which facilitate the FDM design. Frames make possible the representation of the semantic references, as well as the prescriptive facts, of the application environment. Frames allow the representation of the intensional model itself within the FDM. Four types of frames have been described for these representations: attribute, element, inference, and user-view.

The representation of the semantic references of the application environment makes it possible for the FDM to monitor and control the processes which are applied to it. This contributes to achieving a high degree of consistency and reliability. By including the semantic references of the application environment in the FDM itself, it is also possible to fulfill more of the needs of database users. This also decreases the responsibility of the users to understand the organization and structure of the database. For new database models to be accepted by industry they must be as "user friendly" as possible, especially if the model itself is complex.

There are certain areas of current database modeling which have been problematic. These areas have been discussed, both in relation to the FDM and the current models. The issues which have been discussed include multivalued dependencies, multiple user views, the representation of incomplete information, and update and deletion anomalies. In addition, this thesis has also included a discussion of database query issues. Specifically, these issues are the handling of unprocessable queries and query optimization using semantic constraints.

There is no question that the FDM would be a large and complex system. This is an unfortunate by-product of its vast capabilities. However, the future requirements of industry may make this by-product a point of minor significance. Additionally, the optimization of the physical behavior of computer systems continues to improve with time, but is not a concern of the intensional modeling level. The intention of developing the FDM is to provide a design for a complete database system.

BIBLIOGRAPHY

[1] Kent, William. "A Simple Guide to Five Normal Form in Relational Database Theory", Communications of the ACM, Vol.26, No.2 (Feb. 1983) 120-125.

[2] Ullman, Jeffrey D. Principles of Database Systems, 2nd Ed. (Maryland: Computer Science Press, Inc., 1982) 25-32,123-128,212.

[3] Minsky, Marvin. "A Framework for Representing Knowledge", in The Psychology of Computer Vision (P. Winston, Ed.) (New York: McGraw-Hill, 1975) 211-277.

[4] Winston, Patrick Henry. Artificial Intelligence, 2nd Ed. (Massachusetts: Addison-Wesley Publishing Co., 1984) 257-267.

[5] Bobrow, Daniel G. and Terry Winograd. "An Overview of KRL, a Knowledge Representation Language", Cognitive Science, Vol.1, No.1, 3-45.

[6] Goldstein, Ira P. and Bruce Roberts. "Using Frames in Scheduling", in [10], 255-284.

[7] King, Jonathan J., "Intelligent Retrieval Planning", Proc. First National Conference on Artificial Intelligence (Stanford, Ca., 1980) 243-245.

[8] Schank, Roger and Robert Abelson. Scripts Plans Goals and Understanding, (New Jersey: Lawrence Erlbaum Assoc., Inc., 1977) 36-46.

[9] Mylopoulos, John. "An Overview of Knowledge Representation", ACM Workshop on Data Abstraction, Databases, and Conceptual Modeling (1980) 5-12.

[10] Winston, Patrick Henry and Richard Henry Brown. Artificial Intelligence: An MIT Perspective, Vol.1 (Massachusetts: MIT Press, 1982).

[11] Bobrow, Daniel G., Ronald M. Kaplan, Martin Kay, Donald A. Norman, Henry Thompson and Terry Winograd. "GUS, A Frame-Driven Dialog System", Artificial Intelligence, Vol.8, No.1 (1977) 155-173.

[12] Codd, E. F. "A Relational Model of Data for Large Shared Data Banks", Communications of the ACM, Vol.13, No.6 (1970) 377-387.

[13] Codd, E. F. "Further Normalization of the Data Base Relational Model", in Data Base Systems (R. Rustin, Ed.) (1972) 33-64.

[14] Roberts, Bruce and Ira P. Goldstein. The FRL Manual, MIT-AI Lab, Memo 409 (Sept. 1977).

[15] Hammer, Michael and Stanly B. Jr. Zdonik. "Knowledge-Based Query Processing", Proc. Sixth International Conference on Very Large Databases, Montreal (Oct. 1980) 137-147.

[16] King, Jonathan J. "QUIST: A System for Semantic Query Optimization in Relational Databases", Proc. Seventh International Conference on Very Large Data Bases, Cannes, Fr. (Sept. 1981) 510-517.

[17] Hayes, P. J. "The Logic of Frames", in Readings in Artificial Intelligence (B. Webber, N. Nilsson, Ed.) (California: Tioga Publishing Co., 1981) 451-458.

[18] The American Heritage Dictionary of the English Language, (P. Davies, Ed.) (New York: Dell Publishing Co.,Inc., 1975).

[19] Fagin, Ronald. "Multivalued Dependencies and a New Normal form for Relational Databases", ACM Transactions on Database Systems, Vol.2, No.3 (Sept.1977) 262-278.

[20] Swartout, William R. "XPLAIN: A System for Creating and Explaining Expert Systems", Artificial Intelligence, Vol.21, No.3 (Sept.1983) 285-325.

[21] Hammer, Michael and Denis McLeod. "Database Description with SDM: A Semantic Model", ACM Transactions on Database Systems, Vol.6, No.3 (Sept. 1981) 351-386.

[22] Shank, Roger C. and Christofer K. Riesbeck. "Inside Computer Understanding", (New Jersey : Lawrence Erlbaum Assoc., Inc., 1981) 197-227.

[23] King, Jonathan J. "Modeling Concepts for Reasoning about Access to Knowledge", Proceedings of the Workshop on Data Abstraction, Databases and Concept Modelling, Pingree park, Colorado (June 1980) 510-517.

[24] Codd, E. F. "Extending the Database Relational Model to Capture More Meaning", ACM Transactions on Database Systems, Vol.4, No.4 (Dec. 1979) 397-434.

[25] Novak, Gordon S., Jr. "Representation of Knowledge in a Program for Solving Physics Problems", Proc. Fifth International Joint Conference on Artificial Intelligence, Cambridge, Massachusetts (Aug. 1977) 286-291.

[26] Fahlman, Scott E. "Representing Implicit Knowledge", in Parallel Models of Associative Memory (G. E. Henton, F. A. Anderson, Ed.) (New Jersey: Lawrence Erlbaum Assoc., Inc., 1981) 145-159.

[27] Winograd, Terry. "Frame Representations and the Declarative/Procedural Controversy", in Representation and Understanding: Studies in Cognitive Science (D. G. Bobrow, A. Collins, Eds.) (New York: Academic Press, 1975) 185-210.

THE USE OF FRAMES IN DATABASE MODELING

by

BARBARA MOORE SWEET

B. S., Kansas State University, 1982

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1984

ABSTRACT

This thesis introduces a database model based on frames, a conceptual structure developed in Artificial Intelligence. The body of the thesis will argue the importance of developing alternative methods of database modeling, The appropriateness of frames as a structure for database modeling, and describe the benefits available to commercial applications from a frame based model.

The concept of a frame-based database model (FDM) is presented. The ATTRIBUTE, ELEMENT, INFERENCE, and USER-VIEW frame components of the FDM are discussed.

The specific issues of database modeling covered in this thesis are:

- multivalued dependencies
- multiple user views
- representation of incomplete information
- update and deletion anomalies
- unprocessable queries
- query optimization using semantic constraints